



 Universidade Federal do Rio de Janeiro  
 Núcleo de Computação Eletrônica

---




Inteligência Computacional  
**Aplicações – Redes Neurais**  
 2002/1

Anderson Canêdo de Oliveira – acanedo@ufrj.br


**IC – Redes Neurais**


---

- Reconhecimento de padrões
- Extração de características
- Classificação
- Categorização (*Clustering*)
- Métricas
- Problemas
- Redes Neurais MLP
- Estimativa de desempenho


**Reconhecimento de Padrões**


---

- Reconhecimento de Padrões:
  - Categorização (*Clustering*)
  - Classificação
  - Reconhecimento
  - Otimização
  - Aproximação de funções
  - Estimativa, Previsão


**Reconhecimento de Padrões**

---


- Essencial para a sobrevivência dos seres vivos
- Separação de padrões de entrada em grupos ou classes
- Maioria das aplicações de RNAs são de reconhecimento de padrões:
  - Reconhecimento da face humana
  - Reconhecimento de caracteres manuscritos
  - Reconhecimento de voz e locutor
  - Identificação de impressões digitais
  - Previsão de compra ou venda na Bolsa de Valores


**Reconhecimento de Padrões**

---

- Etapas de um sistema de Reconhecimento de padrões:
  - Extração de características
    - Processa dados originais
    - Determina conjunto de características (variáveis mais relevantes)
  - Classificação
    - Recebe vetores de características como entrada
    - Atribui cada vetor a uma das classes  $C_1, C_2, \dots, C_n$

Entrada (dados originais) → Extração de características →  $\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}$  (vetor de características) → Modelo de Classificação → Saída ( $C_1, C_2, \dots, C_n$ )


**Extração de Características**

---


- A utilização direta dos padrões de entrada originais pelo modelo classificador não é muito eficiente
- Características demais pioram o desempenho do classificador

Número de características = Número de variáveis = Dimensionalidade do problema

- Importante: **Selecionar as características que melhor discriminam os padrões para o objetivo da classificação**


Nome, Sexo, Peso, Nota I.C., CR → { Que características usar? } →  $\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}$  (Vetor de características) → Classificador → A, B, C (Classes)

Conjunto de características (Alunos de Informática)



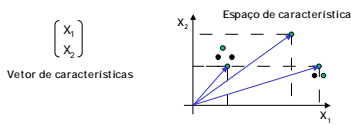
## Extração de Características

- Vetor de características é o conjunto fixo de características dos padrões a serem classificados  
 $X = \text{vetor de características}$  (Ex:  $x_1 = CR$ ,  $x_2 = \text{Nota IC}$ , ...)  
 $d = n^\circ \text{ de características}$
- Problemas:
  - Dados podem conter informações que atrapalham a classificação
  - Variáveis são geralmente correlacionadas
  - Características a serem extraídas dependem do problema (natureza dos padrões)
  - Quais as características que melhor discriminam ou explicam os padrões?



## Extração de Características

- Vetores de características deveriam ser iguais para os padrões de uma mesma classe
- Vetores diferentes para padrões de classes diferentes
- Devem conter informações necessárias para distinguir padrões de classes diferentes
- Padrões ou objetos podem ser representados abstratamente como pontos no espaço de características






## Classificação

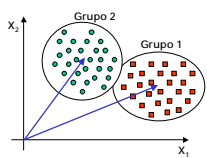
- Classifica o vetor de características em uma das classes já conhecidas
- Aprendizado supervisionado
- Ex: Reconhecimento de dígitos  
10 Classes (1, 2, ..., 0)
- Possíveis classificações:
  - Classificação correta
  - Classificação incorreta
  - Rejeição






## Agrupamento (Clustering)

- Explora semelhanças entre padrões e agrupa os padrões parecidos em categorias ou grupos
- Aprendizado não supervisionado
- Padrões semelhantes ou similares são representados por vetores de características próximos

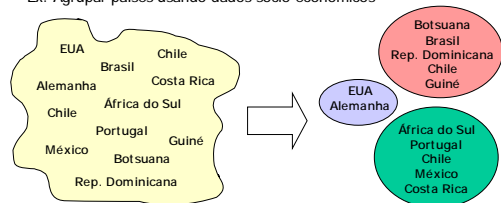





## Agrupamento

- Dependendo do problema fica difícil rotular as categorias
- Se a classe dos padrões de treinamento for conhecida, os grupos podem ser rotulados

Ex: Agrupar países usando dados sócio-econômicos





## Agrupamento

- Métodos mais conhecidos:
  - K-means
  - KNN (K- vizinhos mais próximos)
  - C-means (k-means fuzzy)
- Redes Neurais
  - Rede SOM (Mapas Auto-Organizáveis)
  - ART (Teoria de Ressonância Adaptativa)

## Métricas

- Classificadores de menor distância definem fronteiras de decisão no espaço de padrões
  - Distância de  $X$  para  $P$   $d = \|X - P_i\|$
- Existe mais de uma maneira de definir  $d$
- Métricas calculam similaridade entre padrões no espaço geométrico
- Métricas mais comuns:
  - Distância de Hamming
  - Distância Euclidiana
  - Distância Manhattan
  - Distância quadrática
  - Distância de Mahalanobis

## Problemas

- Problemas linearmente separáveis são aqueles que podem ser satisfeitos utilizando uma reta ou hiperplano como fronteira de decisão.

## Problemas

- Classificação perfeita nem sempre é possível
- Nem todos os problemas são linearmente separáveis
- Difícil validar qual o melhor número de categorias para um determinado problema
- Grande sobreposição de classes (???)
- Causas frequentes de erros:
  - Padrões não podem ser linearmente separados
  - Características podem ser inadequadas para distinguir as diferentes classes
  - Características podem ser muito correlacionadas
  - Podem haver subclasses distintas nos dados

## Redes Neurais - MLP


- Redes de uma camada resolvem apenas problemas linearmente separáveis.
- Solução**
  - Usar redes com mais de uma camada.
- Problema**
  - Como treinar uma rede com mais de uma camada ?

## Redes Neurais - MLP

- MLP - Multilayer Perceptrons**
  - Redes com duas ou mais camadas de neurônios do tipo do Perceptron.
  - Algoritmo de treinamento "Back-propagation error"
    - Proposto por Rumelhart (1986)
    - Fornece um método computacional eficiente para o treinamento de perceptrons de múltiplas camadas
    - Treinamento supervisionado
    - Regra de aprendizagem por correção de erro
    - Generalização do algoritmo LMS (Regra Delta)
    - Baseado no Gradiente Descendente
    - Substituição da função degrau pela função sigmóide
    - Retro-propagação do erro

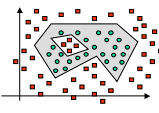
## Redes Neurais - MLP

- Duas etapas:
  - Forward**
    - Ativa a rede a partir das entradas e propaga para as saídas
  - Backward**
    - Utiliza a saída desejada e a saída calculada pela rede para para atualizar os pesos




## Redes Neurais - MLP

- 1ª camada intermediária
  - Traça retas no espaço dos padrões de treinamento
- 2ª camada intermediária
  - Combina as retas traçadas na camada anterior formando regiões convexas
- Camada de saída
  - Cada neurônio forma regiões que são combinações das regiões convexas definidas na camada anterior



- 1 camada intermediária implementa qualquer função contínua
- 2 camadas intermediárias permitem a aproximação de qualquer função



## A regra de aprendizado

Medida de performance

$$E(k) = \frac{1}{2} \sum (y_d - \hat{y})^2$$

Aprendizado

$$\Delta W(k+1) = -\eta \cdot \frac{\partial E(k)}{\partial W(k)}$$

usando a regra da cadeia

$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial e} \cdot \frac{\partial e}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial net} \cdot \frac{\partial net}{\partial W}$$

$\frac{\partial E}{\partial e} = e$      $\frac{\partial e}{\partial \hat{y}} = -1$      $\frac{\partial \hat{y}}{\partial net} = f'(net)$

$\frac{\partial net}{\partial W} = X$  ou  $\hat{y}_i$  saída da camada anterior


Ativação =  $net = \sum w_i x_i = WX$

Propagação =  $sig(x) = \frac{1}{1 + e^{-x}}$

$$\begin{cases} f(net) = sig(net) = \left( \frac{1}{1 + e^{-net}} \right) \\ f'(net) = f(net) \cdot (1 - f(net)) \end{cases}$$

Gradiente local

$$d_j = \frac{\partial E}{\partial net} = \frac{\partial E}{\partial e} \cdot \frac{\partial e}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial net} = -e \cdot f'(net)$$



## A regra de Aprendizado

$$\begin{pmatrix} \text{correção do peso} \\ \Delta W_{ji}(n) \end{pmatrix} = \begin{pmatrix} \text{parâmetro da taxa de aprendizado} \\ \eta \end{pmatrix} \begin{pmatrix} \text{gradiente local} \\ d_j(n) \end{pmatrix} \begin{pmatrix} \text{sinal de entrada do neurônio j} \\ y_c \end{pmatrix}$$

Gradiente local

$$d_j = \frac{\partial E}{\partial net} = \frac{\partial E}{\partial e} \cdot \frac{\partial e}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial net} = -e \cdot f'(net)$$

O gradiente local depende de se o neurônio j é um nó de saída ou se é um nó oculto.


- Se neurônio j é um nó de saída  $d_j = e_j \cdot f'(net)$
- Se neurônio j é um nó oculto  $d_j = f'(net) \cdot \sum_k d_k W_{kj}$



## Redes Neurais - MLP

- Função sigmóide
  - Contínua e derivável
  - $W_0$  determina a posição da sigmóide no eixo das ordenadas
  - Inclinação é determinada pela norma do vetor de pesos  $\|W\|$
- Número de neurônios por camada?
  - Depende do conjunto de treinamento, quantidade de ruído, complexidade da função a ser aprendida.
  - Definido empiricamente
  - Em função das entradas e saídas
  - Nº de conexões 10 vezes menor que o número de exemplos

nº de parâmetros livres = grau de liberdade



## Redes Neurais - MLP

- Atualização dos pesos
  - Por padrão (online) – pesos são atualizados após apresentação de cada padrão.
    - Estável se  $\lambda$  pequeno
    - Geralmente mais rápido
  - Por ciclo (batch) – apresenta todos os padrões de treinamento depois atualiza os pesos.
    - Mais estável
    - Lento se conjunto de treinamento grande e redundante
    - Requer mais memória
    - Estimativa do gradiente mais precisa.



## Redes Neurais - MLP

- Problemas
  - Multimodal – topologias diferentes de Redes Neurais podem resolver os mesmo problema.
  - Deceptivo – topologias semelhantes de 2 redes Neurais podem apresentar comportamentos completamente diferentes.
  - Sem garantia de convergência
    - Pode convergir para mínimos locais
    - Lentidão na convergência
  - Cálculo do erro é preciso apenas para a camada de saída. As camadas intermediárias recebem apenas uma estimativa do erro.

## Redes Neurais - MLP

- **Problemas**
  - Backpropagation é muito lento em superfícies complexas.
  - **Superfície plana (flat spot)** – quando a derivada da sigmoide de um neurônio se aproxima de zero durante o treinamento, o neurônio pode não ter seus pesos ajustados ou ajustados com um valor muito pequeno.
  - **Overfitting** – memoriza padrões de treinamento incluindo suas peculiaridades. Piora a generalização.
  - **Underfitting**
  - **Saturação** – inicialização dos pesos devem ser uniformemente distribuídos dentro de um intervalo pequeno.

## Redes Neurais - MLP

- **Avaliação dos Resultados**
  - Saída binária  $\times$  Saída contínua
  - Saída binária com indecisão (sim, não sei, não)
  - Estratégia “the winner take all”
  - Várias simulações
  - Análise da matriz de confusão
  - Análise estatística dos resultados
    - Média
    - Desvio padrão
  - Validação cruzada
  - Distribuição equilibrada dos padrões no conjunto de treinamento.

## Redes Neurais - MLP

Vantagens	Desvantagens
<ul style="list-style-type: none"> <li>• Solução de problemas não linearmente separáveis</li> <li>• Aprendizado baseado na experiência (por correção de erro)</li> <li>• Generalização</li> <li>• Aplicações:               <ul style="list-style-type: none"> <li>– Classificação</li> <li>– Aproximação de funções</li> </ul> </li> <li>• Paralelismo e robustez</li> </ul>	<ul style="list-style-type: none"> <li>• Dificuldade na definição da arquitetura (nº de camadas, números de neurônios)</li> <li>• Inicialização dos pesos (sorte/azar)</li> <li>• Convergência lenta e incerta</li> <li>• Paralisia do aprendizado (mínimos locais, regiões planas)</li> <li>• Dificuldade na definição dos parâmetros de treinamento</li> </ul>

## Dificuldades de Aprendizado

- **Mínimos Locais**
  - Solução estável, porém não é a melhor solução
  - Estratégias para minorar este problema
    - Taxa de aprendizado decrescente (adaptativa)
    - Adicionar nós intermediários
    - Inclusão do termo momentum
    - Adicionar ruídos aos dados de treinamento
- **Lentidão no treinamento**
  - Utilizar métodos de 2ª ordem
    - Levenberg Maquardt
    - Quase-Newton
    - Gradiente Conjugado
    - Delta-Barra-Delta

## Dificuldades de Aprendizado

- **Overfitting**
  - A rede “decora” os dados de treinamento. Depois de um certo ponto do treinamento a rede piora ao invés de melhorar.
  - Piora a generalização / Memoriza padrões
  - Estratégia para se evitar Overfitting:
    - Encerrar treinamento mais cedo (early stop)

## Estimativa de Desempenho

- Função sigmoide tangente hiperbólica geralmente proporciona um aprendizado mais rápido.
- Pre-processamento dos dados
  - Balanceamento dos dados
  - Normalizar dados (campos individualmente)
- Medida de Desempenho (Taxa de Acerto)
  - **Aparente** - Medida com os dados de treinamento
  - **Verdadeira** - Medida com os dados de teste

$$\text{Taxa de acerto} = \frac{\text{nº de acertos}}{\text{nº de padrões}}$$



## Estimativa de Desempenho

- Hold-out (Split-sample)
  - Técnica mais simples
  - Utiliza uma única partição da amostra
    - Treinamento (1/3)
    - Teste (1/3)
  - Grande quantidade de dados ( $> 1000$ )
    - Aumentar proporção de exemplos de treinamento
  - Pequena quantidade de dados
    - Aproximação pessimista
    - Resultados podem ser imprecisos
    - Utilizar resampling



## Estimativa de Desempenho

- Resampling
  - Várias partições para os conjuntos de treinamento e teste
- Random Subsampling
  - Diferentes partições treinamento/teste/validação escolhidas de forma aleatória
  - Não pode haver interseção entre os conjuntos
  - Performance é calculada para cada partição
  - Performance estimada é a média dos erros para as diferentes partições
  - Permite a obtenção de uma estimativa mais precisa para o desempenho de um modelo.



## Estimativa de Desempenho

- Cross-validation
  - Classe de métodos para estimativa de desempenho verdadeiro
  - K-fold cross-validation
    - Divide o conjunto de dados em K partições mutuamente exclusivas
      - A cada iteração, uma das K partições é usada para testar o modelo. As outras K-1 para treinar
    - Taxa de acerto é a média dos erros das K partições
  - Leaving-one-out
    - N iterações são utilizadas para uma amostra de N.
    - N-fold cross-validation